## AMENDMENTS TO THE SPECIFICATION

Please amend the paragraph immediately following the title "Background of the Invention," on page 2 of the specification, as follows:

As computers have become more widely used and more pervasively networked, information, privacy, and financial losses, etc., due to information security breaches have dramatically increased as well. According to a March 12, 2001 survey press release, published by the Computer Security Institute, eighty-five percent of 538 respondents, primarily large corporations and government agencies, detected computer security breaches in the preceding year. ~~(*http://www.gocsi.com/prelea_000321.htm*, incorporated herein by reference.)~~ Sixty-four percent of the same respondents confirmed financial losses due to computer breaches, with thirty-five percent of 186 respondents being capable of quantifying their losses as totaling $377,828,700. By comparison, the losses from 249 respondents in 2000 totaled $265,589,940, with the average annual total over the prior three year period being $120,240,180. This trend illustrates that the threat of computer crime and other electronic information security breaches continues to grow with financial loss exposure also increasing.

Please amend the first paragraph on page 4 of the specification, as follows:

Several debugging tools, such as, for example, NUMEGA® SOFTICE®, are readily available for this purpose. ~~(http://www.numega.com/drivercentral/components/si_driver.shtml).~~ A debugging tool, more commonly known as a "debugger", is a software development program that allows the user to debug and troubleshoot computer code as the application is being developed. Generally, the debugger acts to interrupt the CPU in order to watch each individual instruction as it arrives for processing and execution. For example, a debugger may be used to analyze and determine precisely which code fragment in a compiled application is responsible for interrogating whether a valid serial number has been provided in order to permit an authorized user to execute the program application. Once the serial number interrogation code fragment is

located by the debugger, it becomes rather simple to decompile the code fragment to determine the characteristics that a valid serial number key must have in order to permit program execution access. For example, for any serial number interrogation code fragment and any valid serial number, the operation of the code fragment $\hat{C}$ on a valid serial number s permits user access (1= True; therefore, allow access) as given by:

Please amend the last paragraph on page 13 of the specification, as follows:

Once the code has been randomly polymorphed, it becomes statistically impossible to retrieve the original application source code. Additionally, each polymorphed copy 21 — 24 of the application randomly differs from any other copy, precluding the possibility of generating a patch or crack for any one polymorphed copy that will work generically with any other polymorphed copy of the application. Moreover, in an alternative exemplary embodiment, the code polymorphing process may be iteratively applied to generate multiple layers of protection by looping 130 the generated code polymorph back through the polymorphic engine as depicted in Fig. 3, such that a layered code polymorph 25 is produced.

Please amend the first full paragraph on page 18 of the specification, as follows:

In yet another exemplary embodiment, a secure output display interface for concealing data may also be used to protect users from applications that may use operating system calls to capture data by placing a hook interface between OS routines and a hardware input device (i.e., from a keyboard, mouse, light pen, etc.). This is generally accomplished by capturing, interpreting and modifying device input before it is made available to other OS processes. Fig. 4 shows a prior art edit textbox 30 which bidirectionally communicates 160 with the operating system 35 and may be directly interrogated to provide 150 the original data from the input device 40 corresponding to the displayed textbox data which is concealed, in this example, by asterisk characters. Fig. 5, however, depicts a system and method in accordance with one embodiment of the present invention, in which the edit textbox 32 is protected 180 from bidirectional communication with the operating system 35. This is generally accomplished with a secure hook-

capture display interface 38 which receives 170 data from an input device 40 and masks 175 that data from the operating system 35.